Project 2

Daniel Fredin, Junhan Li, & Eric Chen

Problem 1

Testing

```r
glass <- read.csv("glass.csv")
```

- response variable will be (Si)

```r
cor(glass[,c("Si", "RI", "Na", "Mg", "Al", "K", "Ca", "Ba", "Fe")])
```

```
##             Si            RI          Na           Mg          Al            K
## Si  1.00000000 -0.5420521997 -0.06980881 -0.165926723 -0.00552372 -0.193330854
## RI -0.54205220  1.0000000000 -0.19188538 -0.122274039 -0.40732603 -0.289832711
## Na -0.06980881 -0.1918853790  1.00000000 -0.273731961  0.15679367 -0.266086504
## Mg -0.16592672 -0.1222740393 -0.27373196  1.000000000 -0.48179851  0.005395667
## Al -0.00552372 -0.4073260341  0.15679367 -0.481798509  1.00000000  0.325958446
## K  -0.19333085 -0.2898327111 -0.26608650  0.005395667  0.32595845  1.000000000
## Ca -0.20873215  0.8104026963 -0.27544249 -0.443750026 -0.25959201 -0.317836155
## Ba -0.10215131 -0.0003860189  0.32660288 -0.492262118  0.47940390 -0.042618059
## Fe -0.09420073  0.1430096093 -0.24134641  0.083059529 -0.07440215 -0.007719049
##            Ca            Ba           Fe
## Si -0.2087322 -0.1021513105 -0.094200731
## RI  0.8104027 -0.0003860189  0.143009609
## Na -0.2754425  0.3266028795 -0.241346411
## Mg -0.4437500 -0.4922621178  0.083059529
## Al -0.2595920  0.4794039017 -0.074402151
## K  -0.3178362 -0.0426180594 -0.007719049
## Ca  1.0000000 -0.1128409671  0.124968219
## Ba -0.1128410  1.0000000000 -0.058691755
## Fe  0.1249682 -0.0586917554  1.000000000
```

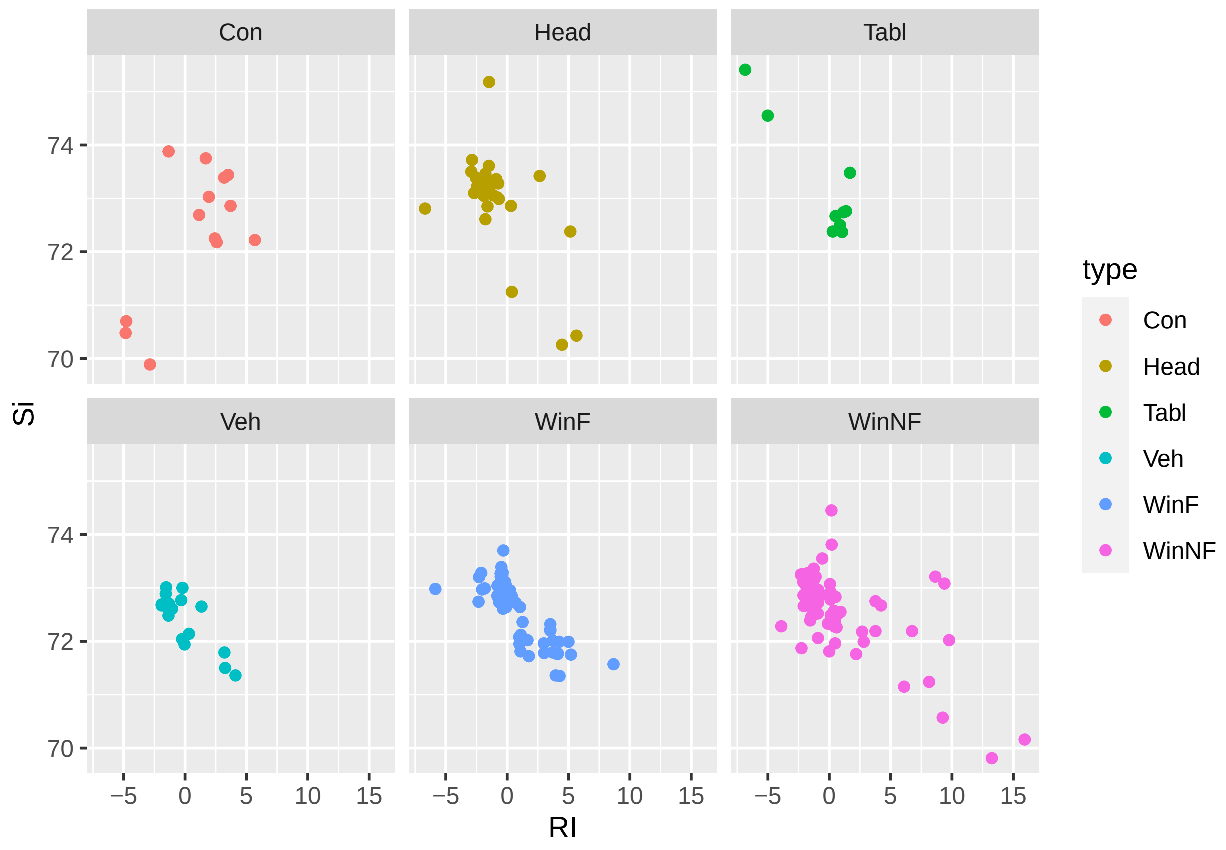
part(a)

- Select four variables of my choice

```
##      Si    RI   Ca type
## 1 71.78  3.01 8.75 WinF
## 2 72.73 -0.39 7.83 WinF
## 3 72.99 -1.82 7.78 WinF
## 4 72.61 -0.34 8.22 WinF
## 5 73.08 -0.58 8.07 WinF
## 6 72.97 -2.04 8.07 WinF
```

- use visualization plot to tell a story, making use the variables selected
- Using color,faceting, theme in ggplot2

```
ggplot(data= glass_new, aes(x=RI, y= Si, color = type)) +
  geom_point()+
  facet_wrap(~type)
```



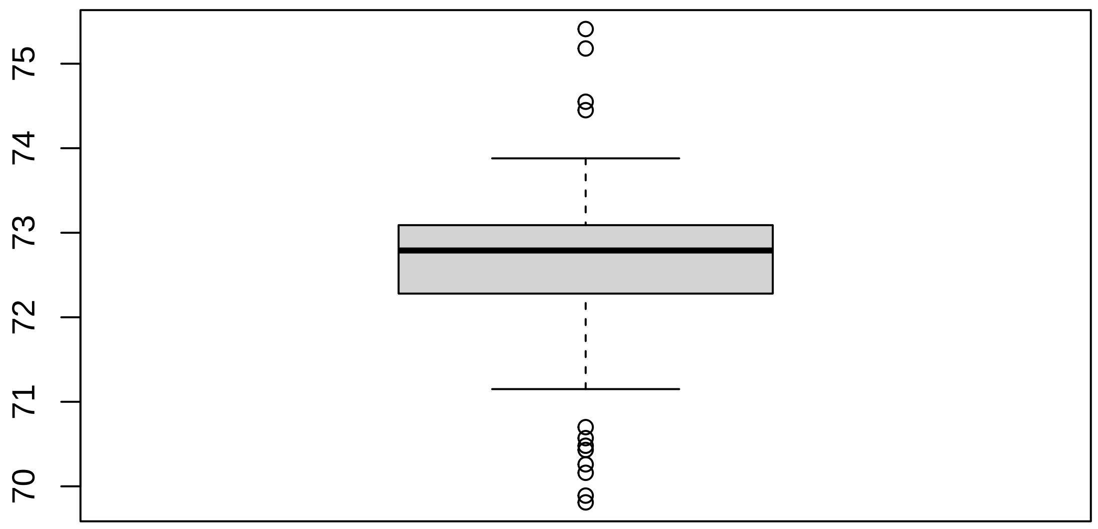
```
# We will first check for the normality of dependent variable
shapiro.test(glass$Si)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  glass$Si
## W = 0.91966, p-value = 2.175e-09
```

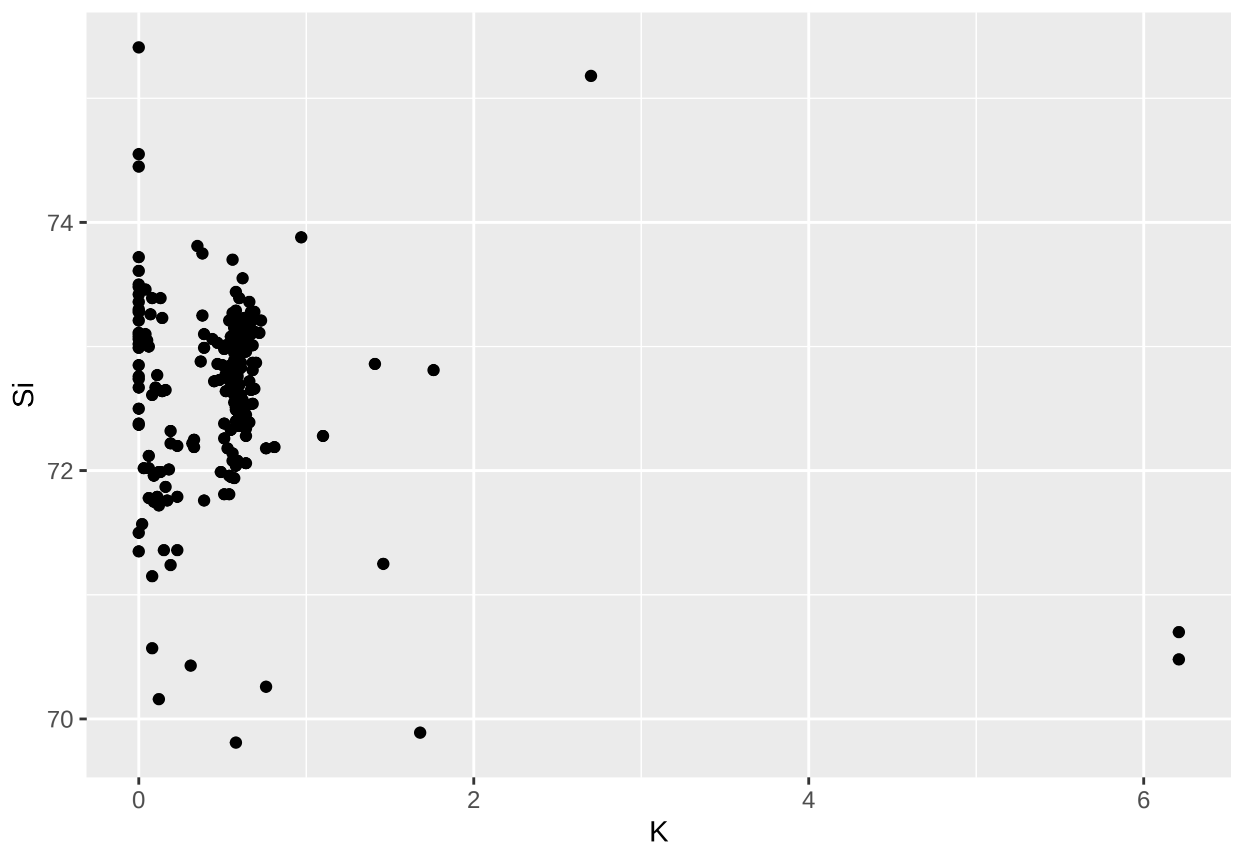
- 2 Paragraphs
  - why we choose variables
  - what you see from graph

Initial assumptions - Dependent variable should be approximately normally distributed - Variables should related to each other linear - use a matrix scatterplot - There should be no significant outliers - Kept at minimum

```
boxplot(glass$Si)
```



```
ggplot(data = glass, aes(x = K, y = Si)) +
  geom_point()
```



Part (b) version 2

(b) Training and testing / k cross-validation method

```
set.seed(123)

glass <- glass %>%
  filter(K<6)

# Regular linear regression with all variables
glass_model1 <- lm(Si ~., data = glass)
predictions <- glass_model1 %>% predict(glass)
model1_results <- data.frame( R2 = R2(predictions, glass$Si),
            RMSE = RMSE(predictions, glass$Si),
            MAE = MAE(predictions, glass$Si))
model1_results
```

```
##          R2       RMSE        MAE
## 1 0.9840169 0.09479197 0.06597926
```

```
summary(glass_model1)
```

```
##
## Call:
## lm(formula = Si ~ ., data = glass)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.72643 -0.04087  0.00933  0.05783  0.17570
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 98.583892   0.467299 210.965  < 2e-16 ***
## RI          -0.006373   0.007302  -0.873  0.38384
## Na          -0.955442   0.016752 -57.034  < 2e-16 ***
## Mg          -0.955069   0.018415 -51.864  < 2e-16 ***
## Al          -0.982915   0.022447 -43.787  < 2e-16 ***
## K           -0.986745   0.031031 -31.799  < 2e-16 ***
## Ca          -0.946004   0.024808 -38.133  < 2e-16 ***
## Ba          -0.969720   0.027175 -35.684  < 2e-16 ***
## Fe          -0.335206   0.072514  -4.623 6.83e-06 ***
## typeHead     0.142240   0.044992   3.161  0.00182 **
## typeTabl     0.055006   0.050478   1.090  0.27717
## typeVeh     -0.038653   0.046831  -0.825  0.41015
## typeWinF     0.024055   0.041356   0.582  0.56146
## typeWinNF   -0.029768   0.037326  -0.798  0.42610
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.09809 on 198 degrees of freedom
## Multiple R-squared:  0.984,  Adjusted R-squared:  0.983
## F-statistic: 937.7 on 13 and 198 DF,  p-value: < 2.2e-16
```

```
 # Regular linear regression with 3 variables
 glass_model2 <- lm(Si  ~ RI + factor(type) + Na , data = glass)
 predictions <- glass_model2 %>% predict(glass)
 results <- data.frame( R2 = R2(predictions, glass$Si),
             RMSE = RMSE(predictions, glass$Si),
             MAE = MAE(predictions, glass$Si))
 results
```

```
##          R2      RMSE       MAE
## 1 0.5274668 0.5154151 0.3281734
```

```
# Training and testing method (SEEMS LIKE OUR BEST MODEL BECAUSE OF LOWER RMSE)
training <- glass$Si %>%
    createDataPartition(p = .75, list = FALSE)
train.data  <- glass[training, ]
test.data <- glass[-training, ]

glass_model3 <- lm(Si ~ Na + Mg + Al + K + Ca + Ba + Fe, data = train.data)
predictions <- glass_model3 %>% predict(test.data)
model3_results <- data.frame( R2 = R2(predictions, test.data$Si),
            RMSE = RMSE(predictions, test.data$Si),
            MAE = MAE(predictions, test.data$Si))
model3_results
```

```
##          R2       RMSE        MAE
## 1 0.9847218 0.08959457 0.07089758
```

```
varImp(glass_model3)
```

```
##      Overall
## Na 54.268549
## Mg 78.410372
## Al 36.533374
## K  29.607760
## Ca 78.560608
## Ba 45.183389
## Fe  3.572403
```

```
# model 4 will use k-cross validation
ctrl <- trainControl(method = "cv", number = 10)
glass_model4 <- train(Si ~ ., data = glass, method = "lm", trControl = ctrl)
print(glass_model4)
```

```
## Linear Regression
##
## 212 samples
##   9 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 190, 190, 191, 190, 191, 191, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.0983153  0.9770289  0.07199174
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
# model 5 will use repeated k-cross validation
ctrl_repeat <- trainControl(method = "repeatedcv", number = 10,  repeats = 3)
glass_model5 <- train(Si ~ ., data = glass,
                      method = "lm", trControl = ctrl_repeat)
print(glass_model5)
```

```
## Linear Regression
##
## 212 samples
##   9 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 3 times)
## Summary of sample sizes: 190, 191, 191, 191, 191, 190, ...
## Resampling results:
##
##   RMSE        Rsquared   MAE
##   0.09818258  0.9805234  0.07183766
```

```
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

Argue your choice - Perform optimization in R, compute the regression coefficients

The lower the value for AIC, the better the fit of the model. The absolute value of the AIC value is not important. It can be positive or negative.

Preconlcusion: Model 2 is a lost better. Based on the RMSE values we generated for the four models above, the model 3 where we used 70% training data model is our best model

Part (c) Use result of best model, answer following questions

(i): Explain coefficient of determination

```
RMSE <- model3_results[1]
RMSE
```

```
##          R2
## 1 0.9847218
```

This coefficient of determination suggests that 98.47% of the variation in Si content could be explained by variables of our best model.

(ii): Find Least-squares estimates: Basically Regression

```
summary(glass_model3)
```

```
##
## Call:
## lm(formula = Si ~ Na + Mg + Al + K + Ca + Ba + Fe, data = train.data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.72798 -0.04789  0.00596  0.06361  0.22488
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 98.82235    0.36017 274.380  < 2e-16 ***
## Na          -0.93971    0.01732 -54.269  < 2e-16 ***
## Mg          -0.99254    0.01266 -78.410  < 2e-16 ***
## Al          -0.99491    0.02723 -36.533  < 2e-16 ***
## K           -0.99553    0.03362 -29.608  < 2e-16 ***
## Ca          -0.98161    0.01249 -78.561  < 2e-16 ***
## Ba          -0.97239    0.02152 -45.183  < 2e-16 ***
## Fe          -0.33551    0.09392  -3.572 0.000474 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.109 on 152 degrees of freedom
## Multiple R-squared:  0.9809, Adjusted R-squared:   0.98
## F-statistic:  1117 on 7 and 152 DF,  p-value: < 2.2e-16
```

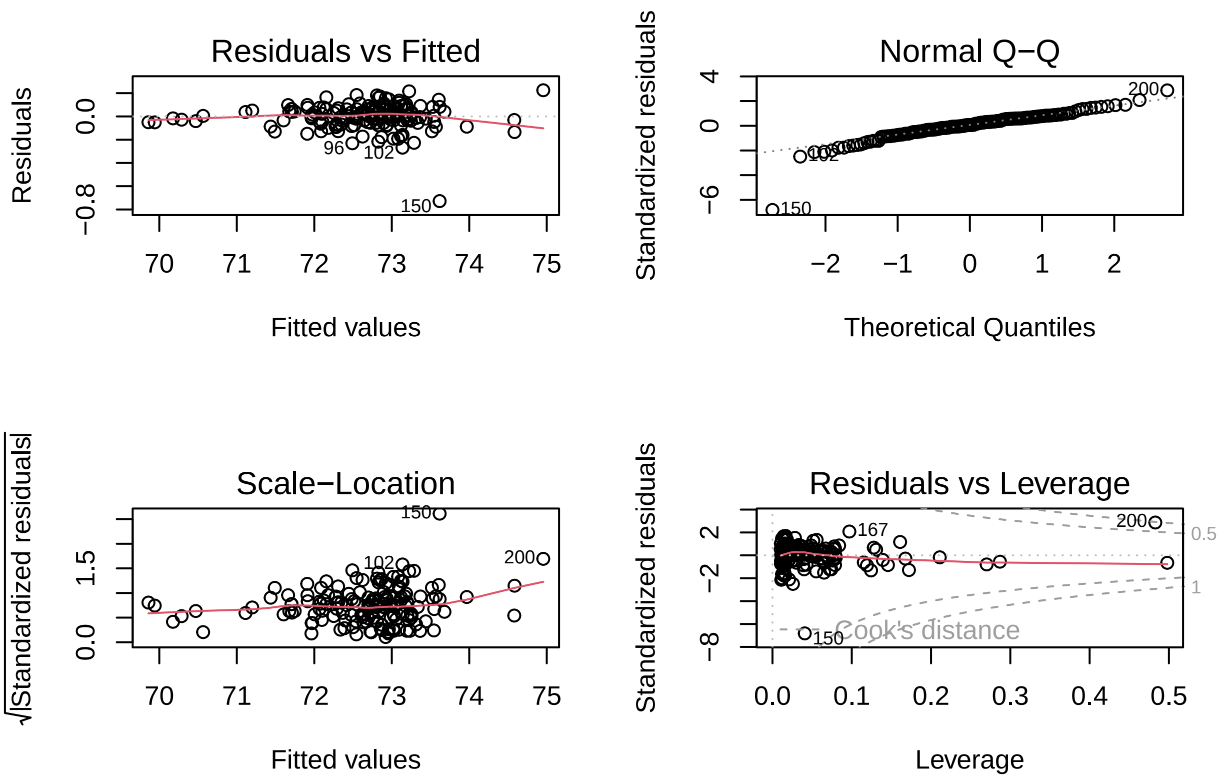
(iii): Interpret slopes and intercept of the problem - determine the significant predictors at 10% level of significance

$$\begin{aligned} Si = 98.81 - (0.94 \cdot Na) - (0.99 \cdot Mg) - (1.00 \cdot Al) \\ - (0.98 \cdot K) - (0.98 \cdot Ca) - (0.96 \cdot Ba) - (0.35 \cdot Fe) \end{aligned}$$

All of the variables in our best model are significant predictors of Si content at 10% confidence. This regression equation suggests that when none of the critical elements are present in the glass, 98.81% of the material will be Si The negative slopes in the equation means that an increase of the presence of any other element by one percent will decrease the Si content of the glass by roughly one percent, which makes sense. However, the exception is Fe, which only decreases Si content by 0.35% for every one percent change.

(iv): Perform Residual analysis - the four tests - four plots group together

```
par(mfrow = c(2,2))
plot(glass_model3)
```



```
# Check for Linearity of glass_model3 using rainbow test
raintest(glass_model3)
```

```
##
##  Rainbow test
##
## data:  glass_model3
## Rain = 0.61458, df1 = 80, df2 = 72, p-value = 0.9828
```

```r
# Check for normality of glass_model3 using shapiro test on residuals
# (This predicts residuals on training data)
real_resid <- residuals(glass_model3)
shapiro.test(real_resid) # violated
```

```
##
##  Shapiro-Wilk normality test
##
## data:  real_resid
## W = 0.8682, p-value = 1.176e-10
```

```r
# (This predicts residuals on TESTING data)
resids <- test.data$Si - predictions # This is what we want!
shapiro.test(resids)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resids
## W = 0.93736, p-value = 0.008762
```

```r
# Check for multicollinearity
vif(glass_model3)
```

```
##       Na       Mg       Al        K       Ca       Ba       Fe
## 2.549561 4.418297 2.421559 2.013984 4.237778 1.821137 1.115514
```

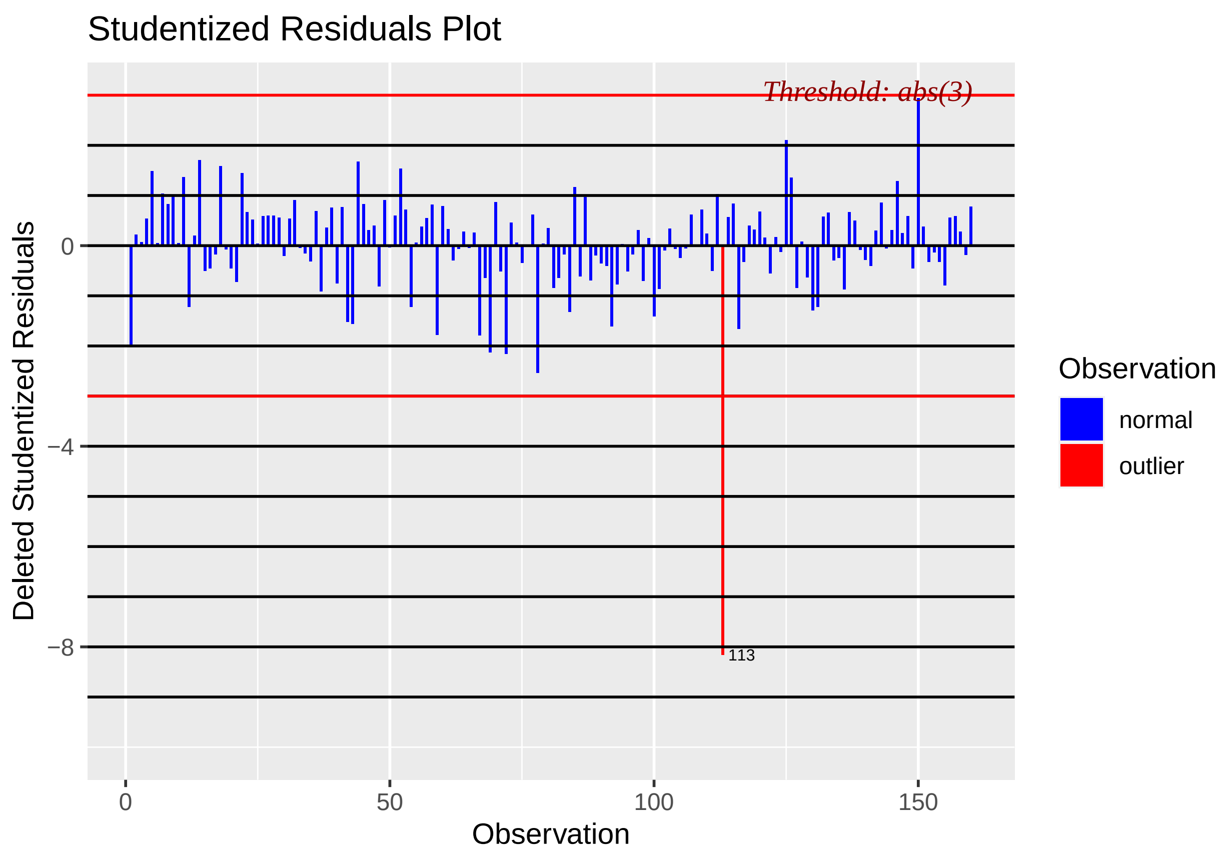
```r
dwtest(glass_model3)
```

```
##
##  Durbin-Watson test
##
## data:  glass_model3
## DW = 2.1194, p-value = 0.7182
## alternative hypothesis: true autocorrelation is greater than 0
```

```r
ols_test_breusch_pagan(glass_model3) # violated
```

```
##
##  Breusch Pagan Test for Heteroskedasticity
##  -----------------------------------------
##  Ho: the variance is constant
##  Ha: the variance is not constant
##
##              Data
##  ------------------------------
##  Response : Si
##  Variables: fitted values of Si
##
##          Test Summary
## -------------------------------
## DF            =    1
## Chi2          =    27.35352
## Prob > Chi2   =    1.694555e-07
```

```
ols_plot_resid_stud(glass_model3) # OUTLIER!!!!
```



Based on the rainbow test on our model, the linearity assumption is not violated, as the null hypothesis of a linear fit is not rejected ($p > 0.05$). However, the results of a Shapiro test on the residuals of the model conclude they are not normal, as $p < 0.05$ here. Heteroscedasticity is also present in the residuals, as is an outlier. Thus, the assumptions on residuals of regression models do not hold.

Part (d): separate observations into

- group 1: type = WinF
- group 2: type= WonNF Variable A1, level alpha = 0.1

```
# Assuming your dataset is named 'data' and the variable of interest is 'Al'
group1 <- subset(glass, type == "WinF")$Al
group2 <- subset(glass, type == "WinNF")$Al

# Perform a t-test: If the distribution of the variable Al follows a normal distribution, use a t-test
ttest_result <- t.test(group1, group2)
ttest_result
```

```
##
##  Welch Two Sample t-test
##
## data:  group1 and group2
## t = -4.9874, df = 143.3, p-value = 1.744e-06
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.3411239 -0.1474776
## sample estimates:
## mean of x mean of y
##  1.163857  1.408158
```

```
p_value <- ttest_result$p.value

# Interpret the t-test results
if (p_value < 0.1) {
  cat("The distributions of the two groups are significantly different at the 0.10 significance level.")
} else {
  cat("The distributions of the two groups are not significantly different at the 0.10 significance leve
}
```

```
## The distributions of the two groups are significantly different at the 0.10 significance level.
```

```
# Perform a non-parametric test: If the distribution of Al is not normal, use a non-parametric test, su

shapiro.test(glass$Al)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  glass$Al
## W = 0.95336, p-value = 2.178e-06
```

```
wilcox_result <- wilcox.test(group1, group2)
wilcox_result
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  group1 and group2
## W = 1399, p-value = 7.875e-07
## alternative hypothesis: true location shift is not equal to 0
```

```
p_value <- wilcox_result$p.value

if (p_value < 0.1) {
  cat("The distributions of the two groups are significantly different at the 0.10 significance level.")
} else {
  cat("The distributions of the two groups are not significantly different at the 0.10 significance leve
}
```

```
## The distributions of the two groups are significantly different at the 0.10 significance level.
```

Part (e):

```
set.seed(123)
# Need 214 values
N <- 214

Na <- rnorm(n=N, mean = mean(glass$Na), sd(glass$Na))
Mg <- rnorm(n=N, mean = mean(glass$Mg), sd(glass$Mg))
Al <- rnorm(n=N, mean = mean(glass$Al), sd(glass$Al))
K <- rnorm(n=N, mean = mean(glass$K), sd(glass$K))
Ca <- rnorm(n=N, mean = mean(glass$Ca), sd(glass$Ca))
Ba <- rnorm(n=N, mean = mean(glass$Ba), sd(glass$Ba))
Fe <- rnorm(n=N, mean = mean(glass$Fe), sd(glass$Fe))

e <- rnorm(n=N, 0, 1)

# Saves coefficients from our model3 as variables
coef <- summary(glass_model3)$coefficients

y <- e + coef[1] + coef[2]*Na + coef[3]*Mg + coef[4]*Al + coef[5]*K + coef[6]*Ca + coef[7]*Ba + coef[8]

summary(y)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   66.26   71.06   72.45   72.58   74.31   80.56
```

```
# Simulated model
sim_model <- lm(y ~ Na + Mg + Al + K + Ca + Ba + Fe)
summary(sim_model)
```

```
##
## Call:
## lm(formula = y ~ Na + Mg + Al + K + Ca + Ba + Fe)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4652 -0.6495 -0.0993  0.7263  3.1087
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 98.31723    1.37528  71.489  < 2e-16 ***
## Na          -0.89469    0.09310  -9.610  < 2e-16 ***
## Mg          -0.98927    0.05028 -19.676  < 2e-16 ***
## Al          -0.99312    0.15116  -6.570 4.04e-10 ***
## K           -1.26263    0.20453  -6.173 3.50e-09 ***
## Ca          -0.98358    0.04853 -20.269  < 2e-16 ***
## Ba          -0.80552    0.15036  -5.357 2.25e-07 ***
## Fe           1.05360    0.73594   1.432    0.154
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 1.035 on 206 degrees of freedom
## Multiple R-squared:  0.8257, Adjusted R-squared:  0.8198
## F-statistic: 139.4 on 7 and 206 DF,  p-value: < 2.2e-16
```

```
# Check for Linearity of glass_model3 using rainbow test
raintest(sim_model)
```

```
##
##  Rainbow test
##
## data:  sim_model
## Rain = 0.81722, df1 = 107, df2 = 99, p-value = 0.8471
```

```
# Check for normality of glass_model3 using shapiro test on residuals
sim_resid <- residuals(sim_model)

shapiro.test(sim_resid) # violated
```

```
##
##  Shapiro-Wilk normality test
##
## data:  sim_resid
## W = 0.99524, p-value = 0.7433
```

```
# Check for multicollinearity
vif(sim_model)
```

```
##       Na       Mg       Al        K       Ca       Ba       Fe
## 1.031273 1.009741 1.010203 1.006343 1.023204 1.036901 1.028267
```

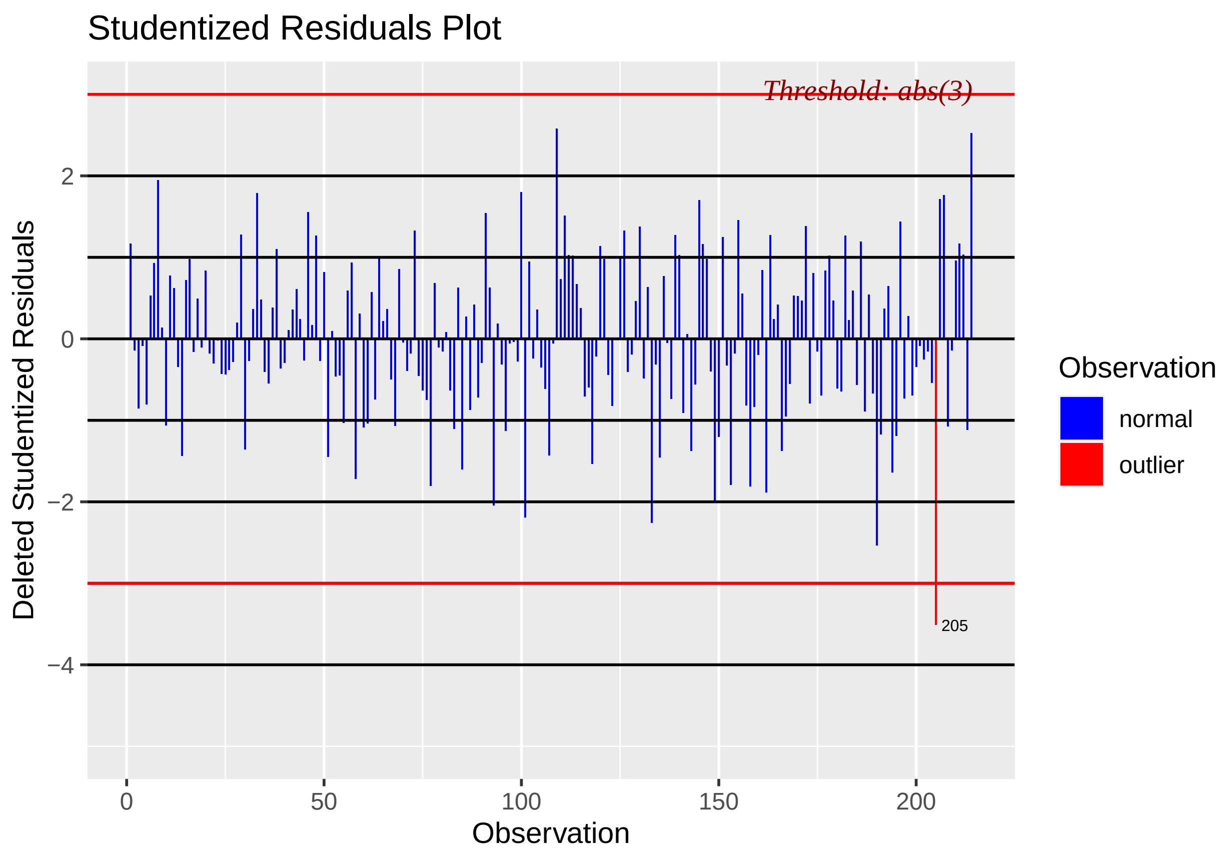
```
dwtest(sim_model)
```

```
##
##  Durbin-Watson test
##
## data:  sim_model
## DW = 1.9962, p-value = 0.4919
## alternative hypothesis: true autocorrelation is greater than 0
```

```
ols_test_breusch_pagan(sim_model) # violated
```

```
##
##  Breusch Pagan Test for Heteroskedasticity
##  -----------------------------------------
##  Ho: the variance is constant
##  Ha: the variance is not constant
##
##             Data
##  -----------------------------
##  Response : y
##  Variables: fitted values of y
##
##         Test Summary
##  ----------------------------
##  DF              =    1
##  Chi2            =    1.339164
##  Prob > Chi2     =    0.2471814
```

```
ols_plot_resid_stud(sim_model) # OUTLIER!!!!
```



Problem 3

a)

i)

We choose to use 10-cross validation. Why we choose it

ii) 10-fold Cross-Validation

Model Accuracy

```
# Predict outcome using model from training data based on testing data
predictions <- predict(cvmodel, newdata=test)

# Create confusion matrix to assess model fit/performance on test data
con_matx <- confusionMatrix(data=predictions, test$COMMIT)
con_matx
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction no yes
##        no  13   4
##        yes  2  11
##
##                Accuracy : 0.8
##                  95% CI : (0.6143, 0.9229)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.0007155
##
##                   Kappa : 0.6
##
##  Mcnemar's Test P-Value : 0.6830914
##
##             Sensitivity : 0.8667
##             Specificity : 0.7333
##          Pos Pred Value : 0.7647
##          Neg Pred Value : 0.8462
##              Prevalence : 0.5000
##          Detection Rate : 0.4333
##    Detection Prevalence : 0.5667
##       Balanced Accuracy : 0.8000
##
##        'Positive' Class : no
##
```

b)

Our objective is to optimize the cost function in order to identify the most economical approach for fulfilling the Christmas order. This entails striving for the lowest cost per day to achieve maximum cost efficiency. By minimizing expenses at each factory, we can ensure a cost-effective process for meeting the demands of the Christmas order.

```
factories <- c("Factory A", "Factory B", "Factory C")
toys <- c("Cars", "Animals", "Robots")

# Define the coefficients of the objective function
costs <- c(1000, 2100, 1500)
```

```r
# Define the constraint matrix
constraint_matrix <- matrix(c(30, 20, 30,
                              40, 50, 10,
                              50, 40, 15), nrow = 3, byrow = TRUE)

# Define the right-hand side of the constraints
constraint_limits <- c(5000, 3000, 2500)

# Set the direction of optimization (minimization)
constraint_directions <- c(">=", ">=", ">=")
direction <- "min"

# Solve the linear programming problem
min_solution <- lp(direction = direction,
               objective.in = costs,
               const.mat = constraint_matrix,
               const.dir = constraint_directions,
               const.rhs = constraint_limits,
               compute.sens=TRUE)


# Check if a solution was found
if (min_solution$status == 0) {
  # Print the optimal solution
  cat("The optimal solution is:\n")
  cat("Factory A:",min_solution$solution[1], "days","\n")
  cat("Factory B:",min_solution$solution[2], "days","\n")
  cat("Factory C:",min_solution$solution[3], "days","\n\n")
  # Print the minimum cost
  cat("The value of the objective function at the optimal solution is:\n")
  cat("Minimum cost: $",min_solution$objval)
} else {
  # No feasible solution found
  print("No feasible solution found.")
}
```

```
## The optimal solution is:
## Factory A: 166.6667 days
## Factory B: 0 days
## Factory C: 0 days
##
## The value of the objective function at the optimal solution is:
## Minimum cost: $ 166666.7
```

The most efficient approach entails running Factory A for approximately 166.67 days, resulting in a minimal cost of $166,666.70, whereas Factory B and Factory C remain non-operational. By adopting this strategy, the company can achieve the most cost-effective outcome.

When considering the optimal solution, it is important to analyze the cost implications of running the factories for different durations. In this scenario, operating Factory for approximately 166.67 days leads to the minimum overall cost. By halting the operations of Factory B and Factory C, the company can avoid additional expenses associated with their functioning.

```r
# Create an empty data frame to store the results
result_df <- data.frame()
factories <- c("Factory A", "Factory B", "Factory C")
toys <- c("Cars", "Animals", "Robots")



# Iterate over the range of 1:3
for (i in 1:3) {

  # Store values for rows and columns in the data frame
  row_df <- data.frame(Factory = factories[i],
                       min_days = min_solution$solution[i],
                       min_cost = min_solution$solution[i]*costs[i])


  result_df <- rbind(result_df, row_df)

}

colnames(result_df) <- c("",
                         "Minimum # of days",
                         "Minimum costs")

# Print the resulting data frame
cat("The optimal solution is:")
```

```
## The optimal solution is:
```

```r
print(result_df)
```

```
##             Minimum # of days Minimum costs
## 1 Factory A          166.6667      166666.7
## 2 Factory B            0.0000           0.0
## 3 Factory C            0.0000           0.0
```

```r
# Sensitivity Analysis

min_solution$sens.coef.from
```

```
## [1]    0.0000  666.6667 1000.0000
```

```r
min_solution$sens.coef.to
```

```
## [1] 1.5e+03 1.0e+30 1.0e+30
```

c)

```r
compute_weibull_stats <- function(shape, scale) {
  # Compute the mean
  mean_value <- scale * gamma(1 + (1/shape))


  # # Compute the median
  # median_value <- scale * qweibull(0.5, shape, scale)


  # Compute the median
  median_value <- scale * (log(2)^(1/shape))


  # Compute the mode
  if (shape > 1)
    mode_value <- scale * ((shape - 1) / shape)^(1/shape)
  else
    mode_value <- 0

  # Compute the variance
  variance_value <- scale^2 * (gamma(1 + (2/shape)) - (gamma(1 + (1/shape)))^2)

  # Return the computed statistics as a named list
  return(list(mean = mean_value,
              median = median_value,
              mode = mode_value,
              variance = variance_value))
}

# lambda = scale
# K = shape

# Example usage
scale_param <- 6
shape_param <- 1

stats <- compute_weibull_stats(shape_param, scale_param)

# Accessing the computed statistics
mean_value <- stats$mean
median_value <- stats$median
mode_value <- stats$mode
variance_value <- stats$variance

# Printing the computed statistics
cat("Mean:", mean_value, "\n")
```

```
## Mean: 6
```

```r
cat("Median:", median_value, "\n")
```

```
## Median: 4.158883
```

```
cat("Mode:", mode_value, "\n")
```

```
## Mode: 0
```

```
cat("Variance:", variance_value, "\n")
```

```
## Variance: 36
```

Maximum likelihood estimator (MLE) of the parameters from the Weibull distribution

Probability Density Function ($f(x;\lambda,k)$) for the Weibull distribution:

$$f(x;\lambda,k) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-\left(\frac{x}{\lambda}\right)^k}, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

The likelihood function ($L_{\hat{x}}(\lambda,k)$):

$$\begin{aligned} L_{\hat{x}}(\lambda,k) &= \prod_{i=1}^{n} \frac{k}{\lambda}\left(\frac{x_i}{\lambda}\right)^{k-1} e^{-\left(\frac{x_i}{\lambda}\right)^k} \\ &= \left(\frac{k}{\lambda}\right)^n \left(\frac{1}{\lambda^{k-1}}\right)^n \left(\prod_{i=1}^{n} x_i^{k-1}\right) \left(e^{-\sum_{i=1}^{n}\left(\frac{x_i}{\lambda}\right)^k}\right) \\ &= \frac{k^n}{\lambda^{nk}} e^{-\sum_{i=1}^{n}\left(\frac{x_i}{\lambda}\right)^k} \prod_{i=1}^{n} x_i^{k-1} \end{aligned}$$

The log-likelihood function ($\ln L_{\hat{x}}(\lambda,k)$):

$$\ln L_{\hat{x}}(\lambda,k) = n\ln(k) - nk\ln(\lambda) - \sum_{i=1}^{n}\left(\frac{x_i}{\lambda}\right)^k + (k-1)\sum_{i=1}^{n}\ln x_i$$

Partial derivative of the log-likelihood function with respect to λ:

$$\frac{\partial \ln L_{\hat{x}}(\lambda,k)}{\partial\lambda} = -\frac{nk}{\lambda} + k\sum_{i=1}^{n}\frac{x_i^k}{\lambda^{k+1}}$$

Solving for the desired parameter of λ by setting the partial derivative equal to zero:

$$
\begin{aligned}
\frac{\partial \ln L_{\hat{x}}(\lambda,k)}{\partial \lambda} &= 0\\
-\frac{nk}{\lambda} + k\sum_{i=1}^{n}\frac{x_i^k}{\lambda^{k+1}} &= 0\\
-\frac{nk}{\lambda} + \frac{k}{\lambda}\sum_{i=1}^{n}\left(\frac{x_i}{\lambda}\right)^k &= 0\\
-n + \sum_{i=1}^{n}\frac{x_i^k}{\lambda^k} &= 0\\
\frac{1}{\lambda^k}\sum_{i=1}^{n}x_i^k &= n\\
\frac{1}{n}\sum_{i=1}^{n}x_i^k &= \lambda^k
\end{aligned}
$$

Therefore the estimator $\hat{\lambda}$ is:

$$
\hat{\lambda} = \left(\frac{1}{n}\sum_{i=1}^{n}x_i^k\right)^{\frac{1}{k}}
$$

Plugging in $\hat{\lambda}$ into the log-likelihood function ($\ln L_{\hat{x}}(\lambda,k)$) and then differentiating with respect to k in order to find the estimator $\hat{k}$:

$$
\begin{aligned}
\frac{\partial \ln L_{\hat{x}}(\lambda,k)}{\partial k} &= \frac{\partial}{\partial k}\left[n\ln k - nk\ln\lambda - \sum_{i=1}^{n}\left(\frac{x_i}{\lambda}\right)^k + (k-1)\sum_{i=1}^{n}\ln x_i\right]\\
&= \frac{\partial}{\partial k}\left[n\ln k - nk\ln\left[\left(\frac{1}{n}\sum_{i=1}^{n}x_i^k\right)^{\frac{1}{k}}\right] - \frac{\sum\limits_{i=1}^{n}x_i^k}{\left[\left(\frac{1}{n}\sum\limits_{i=1}^{n}x_i^k\right)^{\frac{1}{k}}\right]^k} + (k-1)\sum_{i=1}^{n}\ln x_i\right]\\
&= \frac{\partial}{\partial k}\left[n\ln k - \frac{nk}{k}\ln\left(\frac{1}{n}\sum_{i=1}^{n}x_i^k\right) - \frac{\sum\limits_{i=1}^{n}x_i^k}{\left(\frac{1}{n}\sum\limits_{i=1}^{n}x_i^k\right)} + (k-1)\sum_{i=1}^{n}\ln x_i\right]\\
&= \frac{\partial}{\partial k}\left[n\ln k - n\ln\left(\frac{1}{n}\sum_{i=1}^{n}x_i^k\right) - n + (k-1)\sum_{i=1}^{n}\ln x_i\right]\\
&= \frac{n}{k} - \left(\frac{n\sum\limits_{i=1}^{n}x_i^k\ln x_i}{\sum\limits_{i=1}^{n}x_i^k}\right) + \sum_{i=1}^{n}\ln x_i\\
&= \frac{1}{k} - \left(\frac{\sum\limits_{i=1}^{n}x_i^k\ln x_i}{\sum\limits_{i=1}^{n}x_i^k}\right) + \frac{1}{n}\sum_{i=1}^{n}\ln x_i
\end{aligned}
$$

Solving for the desired parameter of k by setting the partial derivative equal to zero:

$$\begin{aligned}
\frac{\partial \ln L_{\hat{x}}(\lambda, k)}{\partial k} &= 0 \\
\frac{1}{k} - \left(\frac{\sum\limits_{i=1}^{n} x_i^k \ln x_i}{\sum\limits_{i=1}^{n} x_i^k} \right) + \frac{1}{n} \sum_{i=1}^{n} \ln x_i &= 0 \\
\left(\frac{\sum\limits_{i=1}^{n} x_i^k \ln x_i}{\sum\limits_{i=1}^{n} x_i^k} \right) - \frac{1}{n} \sum_{i=1}^{n} \ln x_i &= \frac{1}{k}
\end{aligned}$$

Therefore the estimator $\hat{k}$ is:

$$\hat{k} = \left[\frac{\sum\limits_{i=1}^{n} x_i^k \ln x_i}{\sum\limits_{i=1}^{n} x_i^k} - \frac{1}{n} \sum_{i=1}^{n} \ln x_i \right]^{-1}$$